My first ChatBot

Alban Pascal - Paul Hernandez

L1 Int2 - 2023/2024

Table of contents

Introduction	**3**
Objectives	**3**
Functional presentation	**4**
Technical presentation	**5**
Results	**8**
Conclusion	**10**

Introduction

In December, my friend and I created our first ChatBot. It was an interesting project, both in terms of technology and organization.

This project introduced us to artificial intelligence, using text processing algorithms and processes such as the famous TF-IDF method. The project was divided into two parts: first was the development of basic functions such as understanding the content of a file's corpus. The second part aimed at calculating the similarity matrix and generating automatic answers.

We have named our ChatBot "Bobby", because we believe that humanizing our ChatBot makes it more user friendly. Moreover, we designed a graphical interface with the "tkinter" python module, making our chatbot intuitive and accessible for users.

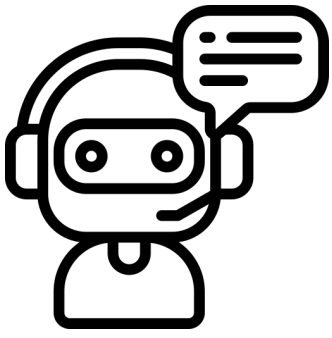
Objectives

The main objective of this project is: Producing a functional ChatBot in Python (in a team of two).

- Improving our knowledge of Python and developing experience with this language
- Learning how to organize and carry out a team project

Our ChatBot must be able to analyze several documents containing texts, and return an appropriate answer to user request.

- Analyze and understand user request
- Search the right document and generate the best possible answer.
- Display the answer appropriately to the user.



Functional presentation

As part of our chatbot project, we had to implement numerous features.

Some were mandatory, but we also chose to add some others, either to make the code and the project easier to understand, or to add some options for the user.

Bobby ChatBot features :

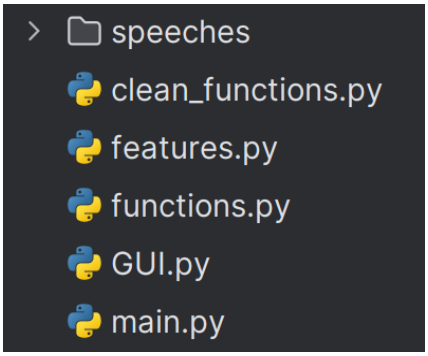
- Mandatory functions of the project:
  - First Part
    - Pre-Function (*Some basics function mainly to cleaned the text files (punctuation and pass upper case to lower case)*)
    - TF-IDF method (*3 functions, TF, IDF and TF-IDF to get the TF-IDF matrix of these files*)
    - 6 features (*The 6 features that we are asked to do*)
  - Second Part
    - Processing the TF-IDF vector from the question
    - Comparing cosine similarity of the question with TF-IDF matrix
    - Generate an “automatic” response

- Other functions added:
  - Function to transform the TF-IDF matrix into a TF-IDF dictionary (*the aim is for the matrix to be human-readable, with the word as the key and its TF-IDF vector as the value*)
  - A unique graphic interface (GUI)
    - Made up of several labels for quick and easy use
    - Chatbot mode to ask any question awaiting an answer from our document corpus
    - List of commands to access the 6 functions in part 1, plus various management functions (*‘clean’ , ‘help’, ‘info’, ‘exit’*)

Technical presentation

Project structure

To better understand our code, we've decided to separate the project into 5 inter-communicating python files, separating functions according to their roles..



- **Speeches folder :** contain all the president's speeches (.txt)
- **Python file**
  - ‘clean_functions’ (*file containing text-cleaning functions*)
  - ‘features’ (*file containing the functions of the 6 features requested in part 1*)
  - ‘functions’ (*file with the largest number of functions, mainly those used to apply the various word processing methods (TF-IDF, question tokenization)*)
  - ‘GUI’ (*file containing the functions used to set up the graphical interface without failure*)
  - ‘main’ (*file to run the program*)

Main algorithms and explanations

The two main algorithms are the **GUI** and the **word processing algorithm** in 'functions.py'.

The 4 functions forming the **GUI** are the following:

- **get_sending() :** is used to retrieve what the user sends from the input frame using the send button
- **printS() :** is used to display what the dire() function returns in the output frame
- **dire() :** this function represents the brain of our project, it's the function that will determine what to answer the user depending on whether he enters a command or a question, this function is implemented with a network of conditions (if, elif, else)
- **dis_gui() :** setup tkinter window with all corresponding frames for proper display

Secondly, the **functions** file represents the flow of the text analyzing process from the implementation of the TF-IDF method to the tokenization of the question until the generation of the chatbot's answer.

Here are the main steps of how this process works and the justification of the data structures used :

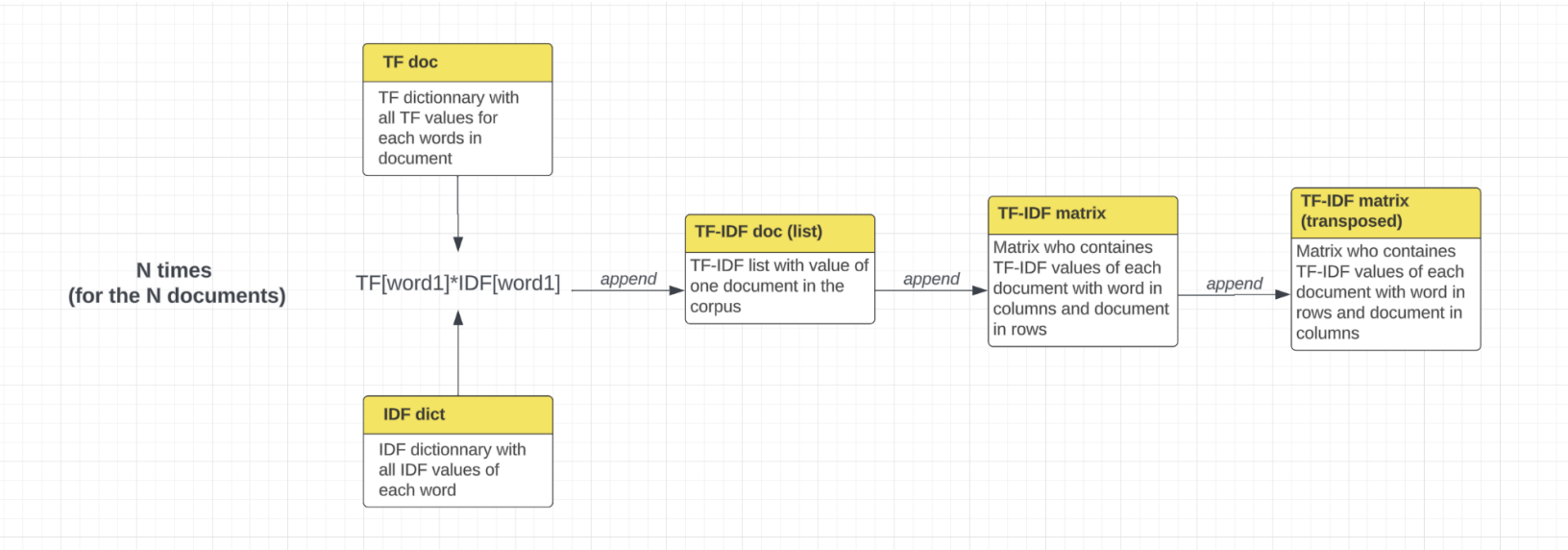
- **TF-IDF Method**

All this project is based on the TF-IDF method, here are the steps we followed:

  - **TF function →** return a dictionary with word in keys and number of repetitions of the word in value from a string (*not a complicated function...*)
  - **IDF function →** this function takes a folder as a parameter, then goes through them one by one in order to calculate the idf value of each word. To do this, we also use a dictionary with the word as key and its tf_idf value as value. We then calculated the frequency of each word in the corpus, using log10 (number of documents/frequency).

$$log_{10}\left(\frac{Total\ number\ of\ documents\ in\ the\ corpus}{Number\ of\ documents\ in\ the\ corpus\ containing\ the\ word}\right)$$

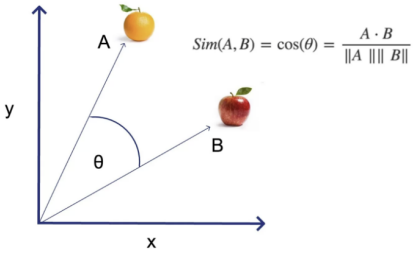
  - **TF-IDF function**



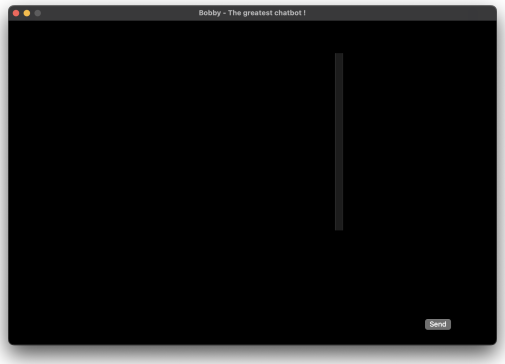
- **Tokenization of the question**

The question tokenization function is **'question_tf_idf()'**, which determines the **TF_IDF value** of each word in the question using the method described above. These values are stored in a list representing a **TF_IDF vector**.

- **Generation automatic answer**
  - **cosine_similarity()** ➜ Returns a decimal value with the following calculation: the scalar product of vectors A and B divided by the product of vectors A and B norms.
  - **most_significant_document()** ➜ For each TF-IDF vector of each document we apply '*cosine_similarity()*' with the TF-IDF vector of the question and determine the largest value found before returning the document in which the maximum is found.
  - **most_important_question_term()** ➜ We browse the TF-IDF values of the question and look for the largest one by returning the associated word.
  - **auto_response()** ➜ Finds the first occurrence of the most significant term returned by *'most_important_question_term()'* in the document returned by *'most_significant_document()'*. Then returns the sentence in which this first occurrence occurs.



Problems and solutions



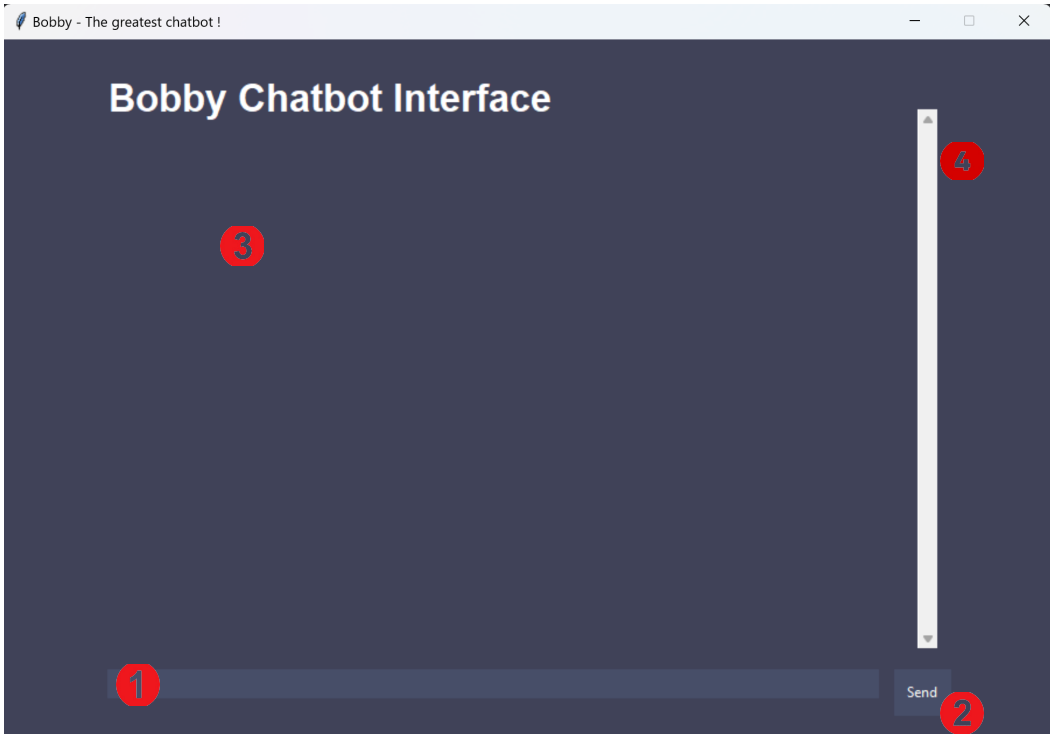
- On **MacOS**, the **GUI display** was a black screen (see below). Unfortunately, this error came from the tkinter module on MacOS, so we couldn't correct it.
- Also, if the user presses “enter” without filling the input label, an error would appear. This error is due to the choice tree (*dire()*) that couldn't process the request.

There was no perfect answer for this problem, we had to review and correct each case separately: *e.g. if the user's question didn't contain any words from the corpus, there was no more important term*.

Results

As for ChatBots in general, Bobby features a Graphical User Interface (GUI) enhancing user experience. When you start the program, a new window is created, and Bobby is ready to answer your queries.

User interface presentation :



1 - Input frame : *Enter the question*

2 - “Send” button : *Send the user's request*

3 - Output frame : *The answer appears below the question*

4 - Scrollbar : *To navigate in the discussion history*

The command menu :

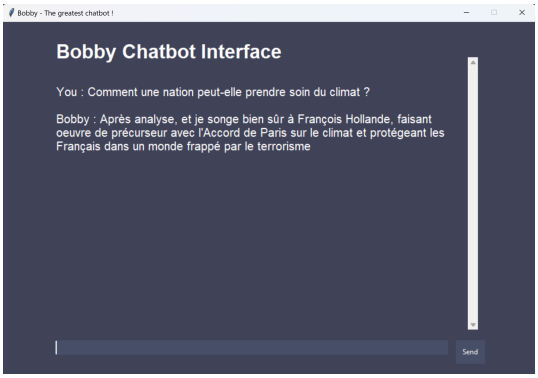
To display the command menu, you need to enter the command '/help' and send it with the button 'send'. This will display the list of valid commands.

Testing a request :

We start by writing a question in the prompt section, here our prompt is "Comment une nation peut-elle prendre soin du climat ?".

After sending our request, Bobby (our ChatBot) sends back the following response :

The response obtained is displayed in the answer frame.



Conclusion

Lessons learned from this project :

- Organizational and communication skills & time management:

We learned how to deal with a due date, the project had to be completed in 3 months, it asked us to organize ourselves and divide the project into multiple tasks to optimize work time.

  - Explain and communicate an idea.
  - Convert an idea into a plan and explain it to your teammate.
  - Openness to critics

- Technical skills in Python

In addition to the mandatory functions we had to use in this project (GIThub), we also worked with new functions to create new features (GUI interface development, chat cleaning function, program stopping function, pitch presentation function)

  - General logic for using python and its features.
  - How functions are used in relation to each other, and what they can be used for.
  - Syntax management for all structures (dictionaries, lists, etc.)